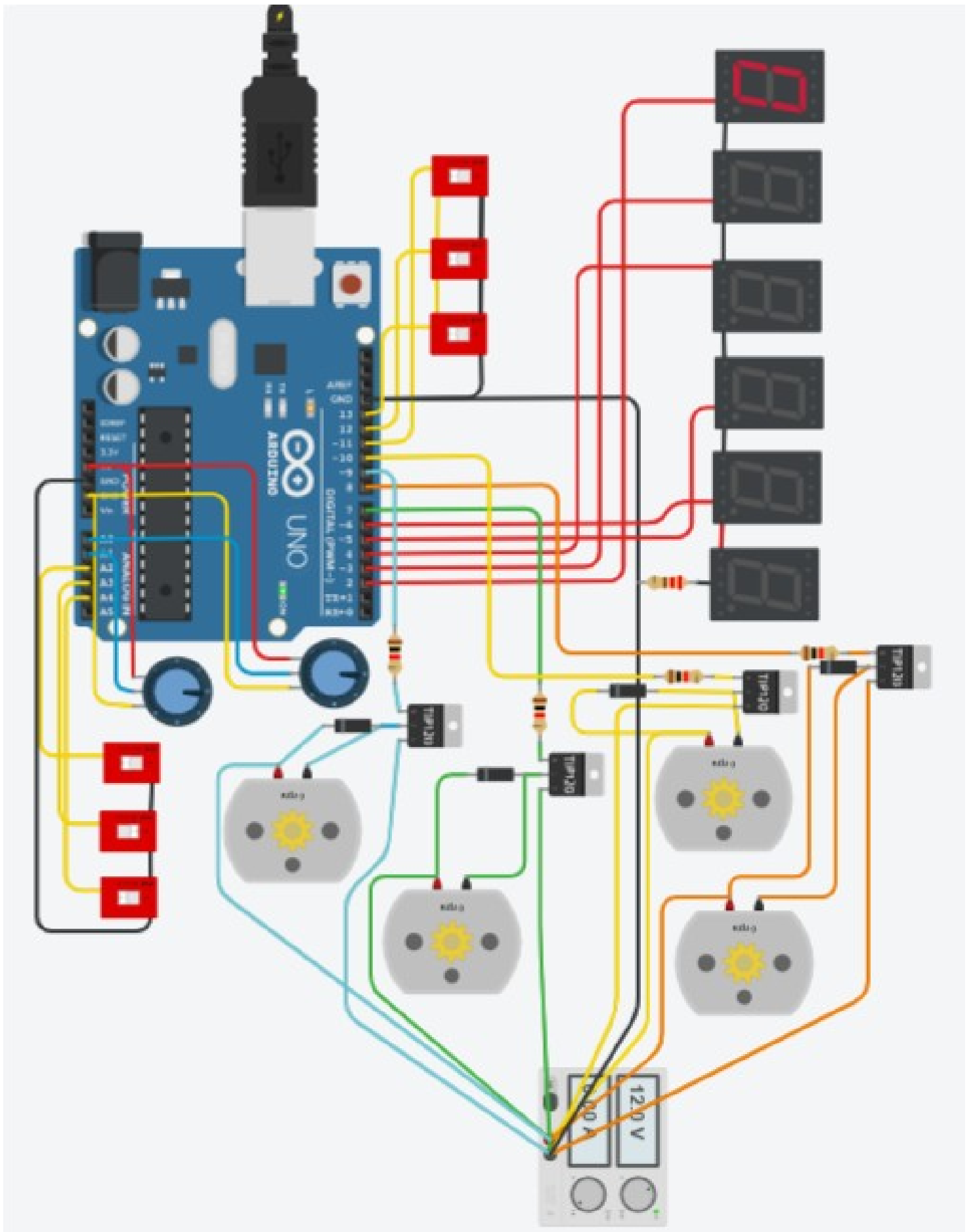


Endereço do projeto simulador: <https://www.tinkercad.com/things/11c4ggls50-fantastic-krunk-wluff/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard>



//2º código fonte com teste

// este código executa teste em todas as saídas quando chave 1 e chave 2 estiverem desligado

// aciona somente a solenoide colocadas na chave, não dá pulso.

// === Definição dos pinos ===

const int led_ch1_verd_n = 2; // LED que indica neutro geral

const int led_ch2_verd_1 = 3; // LED que indica 1ª marcha

const int led_ch2_verd_2 = 4; // LED que indica 2ª marcha

const int led_ch2_verd_3 = 5; // LED que indica 3ª marcha

const int led_ch1_verd_re = 6; // LED que indica ré

const int sol_1 = 7; // Saída para solenoide da 1ª marcha

const int sol_2 = 8; // Saída para solenoide da 2ª marcha

const int sol_3 = 9; // Saída para solenoide da 3ª marcha

const int sol_re = 10; // Saída para solenoide da ré

const int ch2_1 = 11; // Entrada da chave ch_2 saída 1 (seleciona 1ª marcha)

const int ch2_2 = 12; // Entrada da chave ch_2 saída 2 (seleciona 2ª marcha)

const int ch2_3 = 13; // Entrada da chave ch_2 saída 3 (seleciona 3ª marcha)

const int ch1_frente = A2; // Entrada da chave ch_1 posição frente

const int ch1_neutro = A3; // Entrada da chave ch_1 posição neutro

const int ch1_re = A4; // Entrada da chave ch_1 posição ré

// === Estado da marcha ===

int marchaAtual = 0; // Variável que guarda a marcha atual: 0=neutro, 1=1ª, 2=2ª, 3=3ª, 4=ré

```

void setup() {

    // Configura entradas das chaves com resistor pull-up interno

    pinMode(ch2_1, INPUT_PULLUP);

    pinMode(ch2_2, INPUT_PULLUP);

    pinMode(ch2_3, INPUT_PULLUP);

    pinMode(ch1_frente, INPUT_PULLUP);

    pinMode(ch1_neutro, INPUT_PULLUP);

    pinMode(ch1_re, INPUT_PULLUP);


    // Configura saídas dos LEDs

    pinMode(led_ch1_verd_n, OUTPUT);

    pinMode(led_ch2_verd_1, OUTPUT);

    pinMode(led_ch2_verd_2, OUTPUT);

    pinMode(led_ch2_verd_3, OUTPUT);

    pinMode(led_ch1_verd_re, OUTPUT);


    // Configura saídas das solenoides

    pinMode(sol_1, OUTPUT);

    pinMode(sol_2, OUTPUT);

    pinMode(sol_3, OUTPUT);

    pinMode(sol_re, OUTPUT);


    Serial.begin(9600); // Inicia comunicação serial para depuração
}

void loop() {

    // Lê estado das chaves (LOW significa acionada porque usamos INPUT_PULLUP)

```

```

bool ch2_1_state = digitalRead(ch2_1) == LOW;
bool ch2_2_state = digitalRead(ch2_2) == LOW;
bool ch2_3_state = digitalRead(ch2_3) == LOW;
bool ch1_f = digitalRead(ch1_frente) == LOW;
bool ch1_n = digitalRead(ch1_neutro) == LOW;
bool ch1_r = digitalRead(ch1_re) == LOW;

// Caso especial: nenhuma chave acionada → modo teste
if (!ch1_f && !ch1_n && !ch1_r && !ch2_1_state && !ch2_2_state && !ch2_3_state) {
    // Liga todas as saídas (LEDs e solenoides)
    digitalWrite(led_ch1_verd_n, HIGH);
    digitalWrite(led_ch2_verd_1, HIGH);
    digitalWrite(led_ch2_verd_2, HIGH);
    digitalWrite(led_ch2_verd_3, HIGH);
    digitalWrite(led_ch1_verd_re, HIGH);

    digitalWrite(sol_1, HIGH);
    digitalWrite(sol_2, HIGH);
    digitalWrite(sol_3, HIGH);
    digitalWrite(sol_re, HIGH);

} else {
    // Define marchaAtual conforme chaves
    if (ch1_n) {
        marchaAtual = 0; // neutro
    } else if (ch1_r) {
        marchaAtual = 4; // ré
    }
}

```

```
} else if (ch1_f) {  
    if (ch2_1_state) {  
        marchaAtual = 1; // 1ª marcha  
    } else if (ch2_2_state) {  
        marchaAtual = 2; // 2ª marcha  
    } else if (ch2_3_state) {  
        marchaAtual = 3; // 3ª marcha  
    } else {  
        marchaAtual = 0; // neutro  
    }  
}
```

```
// Atualiza LEDs conforme marcha escolhida
```

```
digitalWrite(led_ch1_verd_n, marchaAtual == 0);  
digitalWrite(led_ch2_verd_1, marchaAtual == 1);  
digitalWrite(led_ch2_verd_2, marchaAtual == 2);  
digitalWrite(led_ch2_verd_3, marchaAtual == 3);  
digitalWrite(led_ch1_verd_re, marchaAtual == 4);
```

```
// Mantém ligada a solenoide da marcha escolhida
```

```
digitalWrite(sol_1, marchaAtual == 1);  
digitalWrite(sol_2, marchaAtual == 2);  
digitalWrite(sol_3, marchaAtual == 3);  
digitalWrite(sol_re, marchaAtual == 4);  
}
```

```
delay(100); // Pequena pausa para estabilidade de leitura
```

```
}
```

```
// 1º fonte Original com potenciômetro
```

```
Código fonte Arduino.
```

```
/*
```

```
Projeto: Controle de marchas de trator com Arduino Uno
```

```
Objetivo: Dar um pulso de 5s só uma vez ao trocar a marcha (pré-ação),
```

```
confirmar via potenciômetro e então manter ligada apenas
```

```
a solenoide da marcha selecionada pela chave ch_2.
```

```
Hardware: D0-D13, A0-A5. 2 chaves (ch_1, ch_2), 5 LEDs, 4 solenoides, 2  
potenciômetros.
```

```
*/
```

```
// === Definição dos pinos ===
```

```
const int pot1 = A0;          // Potenciômetro 1 (posição seletora 1) ligado ao pino A0
```

```
const int pot2 = A1;          // Potenciômetro 2 (posição seletora 2) ligado ao pino A1
```

```
const int led_ch1_verd_n = 2; // LED de neutro geral (ch_1 neutro ou ch_2 desligado)
```

```
const int led_ch2_verd_1 = 3; // LED indicador da 1ª marcha
```

```
const int led_ch2_verd_2 = 4; // LED indicador da 2ª marcha
```

```
const int led_ch2_verd_3 = 5; // LED indicador da 3ª marcha
```

```
const int led_ch1_verd_re = 6; // LED indicador de ré
```

```
const int sol_1 = 7;          // Saída da solenoide 1 (marcha 1)
```

```
const int sol_2 = 8;          // Saída da solenoide 2 (marcha 2)
```

```
const int sol_3 = 9;          // Saída da solenoide 3 (marcha 3)
```

```
const int sol_re = 10;        // Saída da solenoide de ré
```

```

const int ch2_1 = 11;          // Entrada da chave ch_2 saída 1 (seleciona 1ª)

const int ch2_2 = 12;          // Entrada da chave ch_2 saída 2 (seleciona 2ª)

const int ch2_3 = 13;          // Entrada da chave ch_2 saída 3 (seleciona 3ª)


const int ch1_frente = A2;     // Entrada da chave ch_1 posição frente (habilita ch_2)

const int ch1_neutro = A3;     // Entrada da chave ch_1 posição neutro (bloqueia ch_2)

const int ch1_re   = A4;       // Entrada da chave ch_1 posição ré (lógica da ré)


// === Limiares dos potenciômetros (ADC) para confirmar neutro ===

const int ADC_NEUTRO_MIN = 291; // Limite mínimo do ADC que representa neutro ( $\approx 1.74$  k $\Omega$ )

const int ADC_NEUTRO_MAX = 580; // Limite máximo do ADC que representa neutro ( $\approx 3.46$  k $\Omega$ )


// === Estado da marcha ===

int marchaAtual = 0;           // Estado atual de marcha: 0=neutro, 1=1ª, 2=2ª, 3=3ª, 4=ré

int marchaAnterior = -1;       // Guarda estado anterior para detectar transição de marcha


// === Função utilitária ===

bool inRange(int v, int lo, int hi) { // Função que verifica se v está na faixa [lo, hi]
    return v >= lo && v <= hi;        // Retorna verdadeiro se dentro da faixa
}


void setup() {                  // Função de configuração inicial

    pinMode(ch2_1, INPUT_PULLUP); // Configura ch_2 saída 1 como entrada com
    pull-up (LOW = ativada)

    pinMode(ch2_2, INPUT_PULLUP); // Configura ch_2 saída 2 como entrada com
    pull-up (LOW = ativada)

    pinMode(ch2_3, INPUT_PULLUP); // Configura ch_2 saída 3 como entrada com
    pull-up (LOW = ativada)

```

```

    pinMode(ch1_frente, INPUT_PULLUP);           // Configura ch_1 frente como entrada com
    pull-up

    pinMode(ch1_neutro, INPUT_PULLUP);           // Configura ch_1 neutro como entrada com
    pull-up

    pinMode(ch1_re, INPUT_PULLUP);               // Configura ch_1 ré como entrada com pull-up


    pinMode(led_ch1_verd_n, OUTPUT);             // Configura LED neutro como saída
    pinMode(led_ch2_verd_1, OUTPUT);             // Configura LED 1ª como saída
    pinMode(led_ch2_verd_2, OUTPUT);             // Configura LED 2ª como saída
    pinMode(led_ch2_verd_3, OUTPUT);             // Configura LED 3ª como saída
    pinMode(led_ch1_verd_re, OUTPUT);            // Configura LED ré como saída


    pinMode(sol_1, OUTPUT);                     // Configura sol_1 como saída
    pinMode(sol_2, OUTPUT);                     // Configura sol_2 como saída
    pinMode(sol_3, OUTPUT);                     // Configura sol_3 como saída
    pinMode(sol_re, OUTPUT);                    // Configura sol_re como saída


    Serial.begin(9600);                         // Inicializa a porta serial para depuração
}

void loop() {                                  // Função principal que roda continuamente

    bool ch2_1_state = digitalRead(ch2_1) == LOW; // Lê se ch_2 saída 1 está ativada (LOW
= chave para GND)

    bool ch2_2_state = digitalRead(ch2_2) == LOW; // Lê se ch_2 saída 2 está ativada
    bool ch2_3_state = digitalRead(ch2_3) == LOW; // Lê se ch_2 saída 3 está ativada
    bool ch1_f = digitalRead(ch1_frente) == LOW; // Lê se ch_1 está na posição frente
    bool ch1_n = digitalRead(ch1_neutro) == LOW; // Lê se ch_1 está na posição neutro
    bool ch1_r = digitalRead(ch1_re) == LOW;      // Lê se ch_1 está na posição ré

```

```
int val1 = analogRead(pot1);           // Lê o valor do potenciômetro 1 (0-1023)
int val2 = analogRead(pot2);           // Lê o valor do potenciômetro 2 (0-1023)
```

```
if (ch1_n) {                            // Se ch_1 está em neutro
    marchaAtual = 0;                     // Define estado neutro
} else if (ch1_r) {                     // Senão, se ch_1 está em ré
    marchaAtual = 4;                     // Define estado ré (será tratado abaixo)
} else if (ch1_f) {                     // Senão, se ch_1 está em frente
    if (ch2_1_state) {                  // Se ch_2 seleciona saída 1
        marchaAtual = 1;                // Define 1ª marcha
    } else if (ch2_2_state) {           // Se ch_2 seleciona saída 2
        marchaAtual = 2;                // Define 2ª marcha
    } else if (ch2_3_state) {           // Se ch_2 seleciona saída 3
        marchaAtual = 3;                // Define 3ª marcha
    } else {                            // Se nenhuma saída de ch_2 ativa (desligado)
        marchaAtual = 0;                // Define neutro
    }
}                                         // Fim da seleção de marcha com base nas chaves
```

```
if (marchaAtual != marchaAnterior) {    // Se houve mudança de marcha (transição detectada)
```

```
    digitalWrite(sol_1, LOW);           // Garante sol_1 desligada antes do pulso
    digitalWrite(sol_2, LOW);           // Garante sol_2 desligada antes do pulso
    digitalWrite(sol_3, LOW);           // Garante sol_3 desligada antes do pulso
    digitalWrite(sol_re, LOW);          // Garante sol_re desligada antes do pulso
```

```
if (marchaAtual == 1) {                 // Caso a nova marcha seja 1ª
```

```

1ª) digitalWrite(sol_3, HIGH);           // Etapa 1: liga sol_3 por 5 segundos (pré-ação da
delay(5000);                             // Mantém o pulso por 5 segundos
digitalWrite(sol_3, LOW);                 // Desliga a solenoide de pré-ação

if (!inRange(val1, ADC_NEUTRO_MIN, ADC_NEUTRO_MAX)) { // Etapa 2: confirma pot_1
em neutro (mn1)
    marchaAtual = 0;                     // Se falhou confirmação, volta para neutro
}

} else if (marchaAtual == 2) {           // Caso a nova marcha seja 2ª
digitalWrite(sol_re, HIGH);              // Etapa 1: liga sol_re por 5 segundos (pré-ação da
2ª) delay(5000);                         // Mantém o pulso por 5 segundos
digitalWrite(sol_re, LOW);               // Desliga a solenoide de pré-ação

if (!inRange(val2, ADC_NEUTRO_MIN, ADC_NEUTRO_MAX)) { // Etapa 2: confirma pot_2
em neutro (mn2)
    marchaAtual = 0;                     // Se falhou confirmação, volta para neutro
}

} else if (marchaAtual == 3) {           // Caso a nova marcha seja 3ª
digitalWrite(sol_1, HIGH);              // Etapa 1: liga sol_1 por 5 segundos (pré-ação da
3ª) delay(5000);                         // Mantém o pulso por 5 segundos
digitalWrite(sol_1, LOW);               // Desliga a solenoide de pré-ação

if (!inRange(val1, ADC_NEUTRO_MIN, ADC_NEUTRO_MAX)) { // Etapa 2: confirma pot_1
em neutro (mn1)
    marchaAtual = 0;                     // Se falhou confirmação, volta para neutro
}

} else if (marchaAtual == 4) {           // Caso a nova marcha seja ré
digitalWrite(sol_2, HIGH);              // Etapa 1: liga sol_2 por 5 segundos (pré-ação da
ré) delay(5000);                         // Mantém o pulso por 5 segundos
digitalWrite(sol_2, LOW);               // Desliga a solenoide de pré-ação

```

```

    if (!inRange(val2, ADC_NEUTRO_MIN, ADC_NEUTRO_MAX)) { // Etapa 2: confirma pot_2
em neutro (mn2)

        marchaAtual = 0;                // Se falhou confirmação, volta para neutro
    }

}

// Fim dos casos de pré-ação

marchaAnterior = marchaAtual;          // Atualiza a marchaAnterior após tratar a
transição

}

// Fim do bloco de transição única

digitalWrite(led_ch1_verd_n, marchaAtual == 0); // Acende LED neutro se estado atual é
neutro

digitalWrite(led_ch2_verd_1, marchaAtual == 1); // Acende LED da 1ª se estado atual é
1ª

digitalWrite(led_ch2_verd_2, marchaAtual == 2); // Acende LED da 2ª se estado atual é
2ª

digitalWrite(led_ch2_verd_3, marchaAtual == 3); // Acende LED da 3ª se estado atual é
3ª

digitalWrite(led_ch1_verd_re, marchaAtual == 4); // Acende LED de ré se estado atual é
ré

digitalWrite(sol_1, marchaAtual == 1);      // Mantém ligada apenas sol_1 na 1ª (Etapa 3:
alimentação final)

digitalWrite(sol_2, marchaAtual == 2);      // Mantém ligada apenas sol_2 na 2ª

digitalWrite(sol_3, marchaAtual == 3);      // Mantém ligada apenas sol_3 na 3ª

digitalWrite(sol_re, marchaAtual == 4);     // Mantém ligada apenas sol_re na ré

delay(100);                                // Pequena pausa para estabilidade de leitura e evitar
bouncing

}

```

Resposta rápida: Você pode copiar o programa do Arduino Uno para um ATmega328P usando um **gravador ISP** (como outro Arduino configurado como “Arduino as ISP” ou um programador USBasp). Para o ATmega328P funcionar de forma “standalone” (sem a placa

Uno), ele precisa de alguns componentes básicos: **crystal de 16 MHz com dois capacitores de 22 pF, resistor de pull-up de 10 kΩ no pino RESET, alimentação de 5 V estável, e conexões de GND e VCC.**

□ Como gravar o programa no ATmega328P

1. Com bootloader já gravado

- o Se o chip já tem o bootloader do Arduino Uno, basta usar um conversor USB-Serial (FTDI, CH340, CP2102) conectado nos pinos **TX, RX, RESET, VCC e GND** do ATmega328P.
- o No Arduino IDE, selecione **Placa: Arduino Uno** e faça o upload normalmente.

2. Sem bootloader (chip virgem)

- o Use outro Arduino Uno como gravador:
 - Carregue o sketch “ArduinoISP” no Uno.
 - Conecte os pinos **MISO, MOSI, SCK, RESET, VCC e GND** do Uno ao ATmega328P.
 - No IDE, escolha **Ferramentas → Gravar Bootloader**.
- o Depois disso, o chip aceita programas via USB-Serial como se fosse um Uno.

3. Usando programador dedicado (USBasp, AVRISP mkII, etc.)

- o Conecte o programador aos pinos ISP do ATmega328P.
- o No Arduino IDE, selecione o programador e use **Upload via Programmer**.

Ótimo, Márcio! Para gravar o programa do Arduino Uno diretamente no microcontrolador **ATmega328P** usando o método **ISP (In-System Programming)**, você precisa conectar os pinos **MISO, MOSI, SCK, RESET, VCC e GND** do Arduino Uno ao ATmega328P conforme a tabela abaixo.

□ Conexões entre Arduino Uno e ATmega328P (modo ISP)

Arduino Uno (como gravador ISP)	ATmega328P (como alvo)
D10 (RESET)	Pin 1 (RESET)
D11 (MOSI)	Pin 17 (MOSI)
D12 (MISO)	Pin 18 (MISO)
D13 (SCK)	Pin 19 (SCK)
5V	Pin 7 (VCC) e Pin 20 (AVCC)
GND	Pin 8 (GND) e Pin 22

□ Componentes adicionais necessários

- **Cristal de 16 MHz** entre os pinos 9 e 10 (XTAL1 e XTAL2)
- **2 capacitores de 22 pF** entre cada pino do cristal e GND
- **Resistor de 10 kΩ** entre RESET (pino 1) e VCC
- **Fonte de 5 V regulada** (pode usar a do próprio Uno)

□ Passos para gravar o programa

1. **Carregue o sketch “ArduinoISP”** no Uno via IDE.
2. Conecte os pinos conforme a tabela acima.
3. No Arduino IDE:
 - o Vá em **Ferramentas → Placa: “ATmega328P” ou “Arduino Uno”**
 - o Selecione **Programador: “Arduino as ISP”**
 - o Clique em **Gravar Bootloader** (se necessário)
 - o Depois, use **Upload via Programador** para gravar o sketch diretamente no chip

Se quiser, posso gerar um **diagrama visual** mostrando essas conexões entre o Uno e o ATmega328P para facilitar sua montagem. Deseja que eu faça isso?

✂ Componentes necessários para o ATmega328P standalone

Para que o microcontrolador funcione fora da placa Arduino Uno, você precisa montar um circuito mínimo:

- **Alimentação:**
 - o VCC (pino 7 e 20) → +5 V
 - o GND (pino 8 e 22) → GND
- **Clock externo:**
 - o Cristal de **16 MHz** entre os pinos 9 (XTAL1) e 10 (XTAL2).
 - o Dois capacitores de **22 pF** entre cada pino do cristal e o GND.

- **Reset:**
 - o Resistor de **10 kΩ** entre RESET (pino 1) e VCC.
 - o Botão opcional entre RESET e GND para reiniciar o chip.
- **Comunicação serial (para gravar ou depurar):**
 - o Pino 2 (RXD) e pino 3 (TXD) conectados a um conversor USB-Serial.
 - o O pino RESET também deve ser ligado ao conversor para permitir upload.
- **LED opcional:**
 - o Um LED com resistor de 220 Ω no pino 19 (PB5, equivalente ao D13 no Uno) para teste (“Blink”).

□ Resumindo

- **Para copiar o programa:** use outro Arduino como ISP ou um programador USBasp para gravar o bootloader e/ou o sketch.
- **Para funcionar standalone:** o ATmega328P precisa de alimentação 5 V, cristal de 16 MHz com capacitores, resistor de 10 kΩ no RESET, e opcionalmente um conversor USB-Serial para comunicação.

□ Assim você transforma o ATmega328P em um “Arduino Uno standalone”, capaz de rodar seus programas sem precisar da placa completa.

Quer que eu monte um **esquema visual de ligação mínima** do ATmega328P standalone (alimentação, cristal, reset e serial) para facilitar a montagem?

sim

Ótimo, Márcio! Vamos montar um **esquema visual de ligação mínima** para usar o **ATmega328P standalone** (fora da placa Arduino Uno), com os componentes necessários para que ele rode o programa copiado do Uno.

⚡ Componentes obrigatórios

- **ATmega328P** (com ou sem bootloader do Arduino Uno)
- **Cristal de 16 MHz**
- **2 capacitores de 22 pF** (para o cristal)
- **Resistor de 10 kΩ** (pull-up no RESET)
- **Fonte de 5 V regulada**
- **Conversor USB-Serial** (FTDI, CH340, CP2102) para gravar programas e depurar via RX/TX
- **Botão opcional** para reset

□ Ligações mínimas

Pino	Conexão necessária
ATmega328P	
7 (VCC)	+5 V
8 (GND)	GND
20 (AVCC)	+5 V
22 (GND)	GND
21 (AREF)	+5 V via capacitor 100 nF (opcional)
9 (XTAL1)	Cristal 16 MHz
10 (XTAL2)	Cristal 16 MHz
XTAL1/XTAL2	Cada pino do cristal ligado a GND via capacitor de 22 pF
1 (RESET)	Resistor 10 kΩ para +5 V; botão opcional para GND
2 (RXD)	Conectar ao TX do conversor USB-Serial
3 (TXD)	Conectar ao RX do conversor USB-Serial
19 (PB5/D13)	LED + resistor 220 Ω para teste (Blink)

□ Passos para gravar o programa

1. Se o chip já tem bootloader

- o Conecte RX/TX/RESET/VCC/GND ao conversor USB-Serial.
- o No Arduino IDE, selecione **Placa: Arduino Uno** e faça upload normalmente.

2. Se o chip não tem bootloader

- o Use outro Arduino Uno como ISP ou um programador USBasp.
- o Grave o bootloader via **Ferramentas → Gravar Bootloader**.
- o Depois, use o conversor USB-Serial para enviar sketches.

□ Esquema simplificado (texto)

Código

ATmega328P

```
+5V ---- VCC (pino 7), AVCC (pino 20)
GND ---- GND (pino 8 e 22)
RESET --[10kΩ]--> +5V
RESET -- botão --> GND
XTAL1 (pino 9) -- Cristal 16MHz -- XTAL2 (pino 10)
XTAL1 --[22pF]--> GND
XTAL2 --[22pF]--> GND
RXD (pino 2) <-- TX do USB-Serial
TXD (pino 3) --> RX do USB-Serial
LED teste: PB5 (pino 19) --[220Ω]--> LED --> GND
```

☐ Assim você tem um **Arduino Uno standalone** usando apenas o ATmega328P e alguns componentes externos.

Quer que eu gere um **diagrama visual ilustrado** mostrando essas conexões (chip + cristal + resistores + fonte + USB-Serial)?

sim

Perfeito, Márcio! Aqui está um **esquema visual simplificado** mostrando como ligar o **ATmega328P standalone** para rodar programas do Arduino Uno.

☐ Ligações mínimas

- **Alimentação**

- VCC (pino 7 e 20) → +5 V
- GND (pino 8 e 22) → GND
- AREF (pino 21) → capacitor de 100 nF para GND (opcional, estabiliza referência)

- **Reset**

- RESET (pino 1) → resistor de 10 kΩ para +5 V
- Botão opcional entre RESET e GND

- **Clock**

- Cristal de 16 MHz entre pinos 9 (XTAL1) e 10 (XTAL2)
- Dois capacitores de 22 pF entre cada pino do cristal e GND

- **Serial (para gravar e depurar)**

- Pino 2 (RXD) → TX do conversor USB-Serial
- Pino 3 (TXD) → RX do conversor USB-Serial
- RESET também ligado ao conversor para permitir upload

- **Teste (Blink)**

- LED + resistor de 220 Ω no pino 19 (PB5, equivalente ao D13 no Uno)

☐ Esquema em texto

Código

ATmega328P

```
[1] RESET ----[10kΩ]---- +5V
      |-- botão --> GND
[7] VCC ----- +5V
[8] GND ----- GND
[20] AVCC ----- +5V
```

```

[22] GND ----- GND
[21] AREF ----[100nF]---- GND (opcional)

[9] XTAL1 ---||--- Cristal 16MHz ---||--- [10] XTAL2
           |           |
           [22pF]       [22pF]
           |           |
          GND          GND

```

```

[2] RXD <---- TX do USB-Serial
[3] TXD ----> RX do USB-Serial
RESET -----> DTR do USB-Serial (via capacitor 100nF)

```

```

[19] PB5/D13 ----[220Ω]----> LED ----> GND

```

➤ Como gravar o programa

1. Se o chip já tem bootloader:

- o Conecte RX/TX/RESET/VCC/GND ao conversor USB-Serial.
- o No Arduino IDE, selecione **Placa: Arduino Uno** e faça upload.

2. Se não tem bootloader:

- o Use outro Arduino como ISP ou um programador USBasp.
- o Grave o bootloader via **Ferramentas → Gravar Bootloader**.
- o Depois, use o conversor USB-Serial para enviar sketches.

☐ Esse é o circuito mínimo para transformar o **ATmega328P** em um **Arduino Uno standalone**.

Você gostaria que eu gere uma **ilustração gráfica** mostrando o chip com os componentes (cristal, resistores, capacitores, fonte e USB-Serial) conectados?

Para gravar um ATmega328P utilizando um conversor FTDI que não possui pinos DTR ou RTS, a solução é realizar o **reset manual do microcontrolador** no momento exato em que a gravação se inicia. Outra alternativa é usar um gravador diferente, como um USBasp ou outro Arduino configurado como programador ISP.

Método 1: Reset Manual (com FTDI sem DTR/RTS)

O processo de gravação do Arduino via serial depende de um pulso automático na linha de reset para iniciar o bootloader. Sem os pinos DTR ou RTS para fazer isso automaticamente, você precisará intervir manualmente:

1. **Conexões:** Conecte o FTDI ao ATmega328P da seguinte forma, garantindo que o chip esteja em um circuito standalone funcional (com cristal oscilador, capacitores e resistor de pull-up no reset, se aplicável):
 - o FTDI TX para ATmega RX (Pino 3, PD1)

- o FTDI RX para ATmega TX (Pino 2, PD0)
- o FTDI VCC para ATmega VCC e AVCC (Pinos 7, 20)
- o FTDI GND para ATmega GND e AGND (Pinos 8, 22)
- o Conecte um fio ou botão de reset ao pino de **Reset** do ATmega (Pino 1, PC6).
- **Preparação na IDE:**
 - o Abra a IDE do Arduino.
 - o Selecione a placa correta (**Arduino Uno** é o mais comum para o 328P) em Ferramentas > Placa .
 - o Selecione a porta serial correta do seu conversor FTDI em Ferramentas > Porta Serial .
 - o Tenha o código pronto para carregar.
- **O Momento do Reset:**
 - o Clique no botão **Carregar** (Upload) na IDE do Arduino.
 - o Observe a barra de status na parte inferior da IDE.
 - o Quando a mensagem "Compilando sketch..." mudar para "Carregando..." (ou "Uploading..."), pressione e solte imediatamente o botão de reset (ou toque o fio de reset no GND por um instante).
 - o Pode ser necessário tentar algumas vezes para pegar o *timing* certo. A janela de tempo é curta, logo após a compilação terminar e o upload começar.

Método 2: Usando Outro Gravador (Mais Confiável)

Se o método manual for muito difícil, a maneira mais confiável e recomendada é usar um programador AVR dedicado ou outro Arduino como um [gravador ISP \(In-System Programming\)](#).

- **Usando outro Arduino como ISP:** Você pode carregar o sketch "ArduinoISP" em uma placa Arduino Uno/Nano e usá-la para programar o ATmega328P diretamente via pinos ICSP (MISO, MOSI, SCK, Reset, VCC, GND).
- **Gravador USBasp:** Um gravador USBasp é uma ferramenta de baixo custo e muito eficaz para programar microcontroladores AVR sem depender da interface serial/bootloader.

Esses métodos de programação ISP não requerem os pinos DTR/RTS do FTDI, pois usam uma interface de comunicação e um método de gravação completamente diferentes.

ATmega328

PCINT15/RESET	1		28	ADC5/SCL/PCINT13
PCINT16/RXD	2		27	ADC4/SDA/PCINT12
PCINT17/TXD	3		26	ADC3/PCINT11
PCINT18/INT0	4		25	ADC2/PCINT10
PCINT19/OC2B/INT1	5		24	ADC1/PCINT9
PCINT20/XCK/T0	6		23	ADC0/PCINT8
VCC	7		22	GND
GND	8		21	AREF
PCINT6/XTAL1/OSC1	9		20	AVCC
PCINT7/XTAL2/OSC2	10		19	SCK/PCINT5
PCINT21/OC0B/T1	11		18	MISO/PCINT4
PCINT22/OC0A/AIN0	12		17	MISO/OC2A/PCINT3
PCINT23/AIN1	13		16	SS/OC1B/PCINT2
PCINT0/CLK0/ICP1	14		15	OC1A/PCINT1

ATmega328

DroneBotWorkshop.com



ATmega328

RESET	1		28	ANALOG INPUT 5
DIGITAL PIN 0 (RX)	2		27	ANALOG INPUT 4
DIGITAL PIN 1 (TX)	3		26	ANALOG INPUT 3
DIGITAL PIN 2	4		25	ANALOG INPUT 2
DIGITAL PIN 3 (PWM)	5		24	ANALOG INPUT 1
DIGITAL PIN 4	6		23	ANALOG INPUT 0
VCC	7		22	GND
GND	8		21	ANALOG REFERENCE
CRYSTAL 1	9		20	VCC
CRYSTAL 2	10		19	DIGITAL PIN 13
DIGITAL PIN 5 (PWM)	11		18	DIGITAL PIN 12
DIGITAL PIN 6 (PWM)	12		17	DIGITAL PIN 11 (PWM)
DIGITAL PIN 7	13		16	DIGITAL PIN 10 (PWM)
DIGITAL PIN 8	14		15	DIGITAL PIN 9 (PWM)

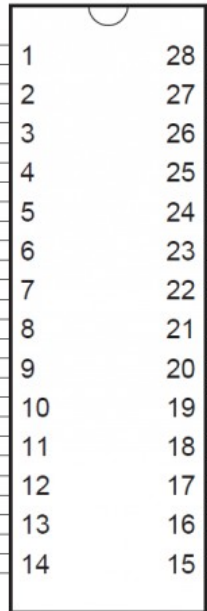
ATmega328

DroneBotWorkshop.com



Arduino Pin

RESET (PCINT14/RESET) PC6 □ 1
digital 0 (RX) (PCINT16/RXD) PD0 □ 2
digital 1 (TX) (PCINT17/TXD) PD1 □ 3
digital 2 (PCINT18/INT0) PD2 □ 4
digital 3 (PCINT19/OC2B/INT1) PD3 □ 5
digital 4 (PCINT20/XCK/T0) PD4 □ 6
VCC VCC □ 7
GND GND □ 8
Clock (PCINT6/XTAL1/TOSC1) PB6 □ 9
Clock (PCINT7/XTAL2/TOSC2) PB7 □ 10
digital 5 (PCINT21/OC0B/T1) PD5 □ 11
digital 6 (PCINT22/OC0A/AIN0) PD6 □ 12
digital 7 (PCINT23/AIN1) PD7 □ 13
digital 8 (PCINT0/CLKO/ICP1) PB0 □ 14



28 □ PC5 (ADC5/SCL/PCINT13)
27 □ PC4 (ADC4/SDA/PCINT12)
26 □ PC3 (ADC3/PCINT11)
25 □ PC2 (ADC2/PCINT10)
24 □ PC1 (ADC1/PCINT9)
23 □ PC0 (ADC0/PCINT8)
22 □ GND
21 □ AREF
20 □ AVCC
19 □ PB5 (SCK/PCINT5)
18 □ PB4 (MISO/PCINT4)
17 □ PB3 (MOSI/OC2A/PCINT3)
16 □ PB2 ($\overline{SS}$ /OC1B/PCINT2)
15 □ PB1 (OC1A/PCINT1)

Arduino Pin

analog 5
analog 4
analog 3
analog 2
analog 1
analog 0
GND
analog reference
VCC
digital 13
digital 12
digital 11
digital 10
digital 9

Logica do programa e código fonte

Criar um projeto para arduino uno que use somente as portas d0 a d13 e a0 até a5 disponíveis e crie o código fonte que controle marchas de um trator, com 2 chaves ch_1 e ch_2 que aciona 5 leds e 4 solenoides e 2 potenciômetros que informam a posição da seletora de marcha pot_1 e pot_2.

Ha 5 leds verdes:

led_ch1_verd_n,
led_ch2_verd_1,
led_ch2_verd_2,
led_ch2_verd_3,
led_ch1_verd_re,

Há 2 chaves ch_2 e ch_1 que conforme posição colocada da chave fara a mudança de marcha:

Chave 2 = ch_2, esta chave seletora tem 3 saídas = saida_1, saida_2, saida_3 e desligada:

saida_1: Quando saida_1 ativado ligar , sol_1 e led_ch2_verd_1,

saida_2: Quando saida_2 ativado ligar sol_2 e led_ch2_verd_2,

saida_3: Quando saida_3 ativa, ligar sol_3 e led_ch2_verd_3,

desligado = se nenhuma saída está ligada = ativa, ligar:
led_ch2_verd_n, somente esta opção libera o uso da ch1_re.

Ha uma chave chamada ch_1 que controla 3 ações:

frente = ch1_frente, somente esta opção ativada permite usar a ch_2

neutro = ch1_neutro, aciona led_ch1_verd_n e não permite usar a ch_2

Ré = ch1_re, aciona led_ch1_verd_re e sol_re

Há 4 solenoide de 12 volts chamadas:

sol_1,

sol_2,

sol_3,

sol_re

2 potenciômetros com as mesmas características com 3 fios: vermelho, azul e amarelo, na escala de 200k que apresenta as medidas: ele inicia com as medidas do vermelho e azul em 1.5k vai até 5.2k, azul e amarelo fica com leitura fixa de 4.1k e azul e amarelo inicia em 5.0k vai até 1.0k.

Há 2 potenciômetros chamados pot_1 e pot_2 que informam o estado atual da marcha:

O Potenciômetro 1 = pot_1 informa:

3ª marcha = Entre 1.5 kΩ a 1.73kΩ = posição ativa em pot_m3

Neutro 1 = Entre 1.74kΩ a 3.46kΩ = posição ativa em pot_mn1

1ª Marcha = Entre 3.47kΩ e 5.2kΩ = posição ativa em pot_m1

O potenciômetro 2 = pot_2 informa:

2ª marcha = Entre 1.5 kΩ a 1.73kΩ = posição ativa em pot_m2

Neutro 2 = Entre 1.74kΩ a 3.46kΩ = posição ativa em pot_mn2

Ré = Entre 3.47kΩ e 5.2kΩ = posição ativa em pot_mre

Para que seja feito a troca de marcha tem que ser feito em 3 etapas:

1ª tem que ter uma alimentação temporária de uma solenoide por 5 segundos

2ª confirmação de posição do potenciômetro

3ª alimentação da solenoide selecionada conforme marcha escolhida em chave 2 = ch_2:

Procedimento de troca de marcha combinação de solenoides:

1ª marcha = ligar solenoide sol_3 por 5 segundos, verificar posição de pot_1 se está em pot_mn1 estando ok ligar, sol_1.

2ª marcha = ligar solenoide sol_re por 5 segundos, verificar posição de pot_2 se está em pot_mn2 estando ok, ligar sol_2.

3ª marcha = ligar solenoide sol_1 por 5 segundos, verificar posição de pot_1 se está em pot_mn2 estando ok, ligar sol_3.

Marcha Ré = ligar solenoide sol_2 por 5 segundos, verificar posição de pot_2 se está em pot_mn2 estando ok, ligar sol_re.